Specifications and FPGA implementation of a systol Hopfield-type associative memory

Mihu, I.Z. Brad, R. Breazu, M.
Dept. of Comput. Sci., Sibiu Univ., Romania;

This paper appears in: **Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on**

Meeting Date: 07/15/2001 - 07/19/2001
Publication Date: 15-19 July 2001
Location: Washington, DC USA
On page(s): 228 - 233 vol.1
Volume: 1
Reference Cited: 14
Number of Pages: 4 vol. xlvi+3014
Inspec Accession Number: 7023382

Abstract:
Neural **networks** are non-**linear** static or dynamical systems that learn to so problems from examples. Most of the learning algorithms require a lot of com power and, therefore, could **benefit** from fast dedicated hardware. One of the common architectures used for this special-purpose hardware is the systolic a design and implementation of different neural **network** architectures in systo can be complex, however. The paper shows the manner in which the Hopfield **network** can be mapped into a 2-D systolic array and presents an FPGA impl of the proposed 2-D systolic array

Index Terms:
Hopfield neural nets content-addressable storage field programmable gate arrays n systolic arrays 2D systolic array FPGA implementation learning algorithms neural architectures special-purpose hardware specifications systolic Hopfield-type associ memory

Documents that cite this document
There are no citing documents available in IEEE Xplore at this time.

Search Results [PDF FULL-TEXT 384 KB] DOWNLOAD CITATION

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

Specifications and FPGA Implementation of a Systolic Hopfield-type Associative Memory

Ioan Z. MIHU
"Lucian Blaga" University of Sibiu
Computer Science Department
Emil Cioran 4, 2400 Sibiu
Romania
mihuz@vectra.sibiu.ro

Remus BRAD
"Lucian Blaga" University of Sibiu
Computer Science Department
Emil Cioran 4, 2400 Sibiu
Romania
rbrad@jupiter.sibiu.ro

Macarie BREAZU
"Lucian Blaga" University of Sibiu
Computer Science Department
Emil Cioran 4, 2400 Sibiu
Romania
mac@vectra.sibiu.ro

Abstract

Neural Networks are non-linear static or dynamical systems that learn to solve problems from examples. Most of the learning algorithms require a lot of computing power and, therefore, could benefit from fast dedicate hardware. One of the most common architectures used for this special – purpose hardware is the Systolic Array [9]. The design and implementation of different Neural Network architectures in Systolic Arrays can be complex, however. This paper shows the manner in which the Hopfield Neural Network can be mapped into a 2-D Systolic Array and present an FPGA implementation of the proposed 2-D Systolic Array.

1 Introduction

Neural Networks (NNs) imply a requirement for massive fine-grained parallelism, with very high levels of interconnection and simple processing at each node. It is from this parallelism that they derive their power.

The intrinsically parallel structure of NNs maps poorly on to conventional Von Neumann computer architectures. Therefore, these architectures are not suitable for implementing real-time NN systems. The mismatch between the parallelism required for NNs and the performance of sequential computer architectures is exacerbated as networks increase in size. As a consequence there is much research activity in the implementation area, investigating technologies and architectures that allow the parallelism of NNs to be mapped into hardware.

A number of technologies have been proposed for the implementation of neural net systems, including analogue very large scale integration (VLSI), digital VLSI, opto-electronics and optical technologies. Each of these has some advantages and disadvantages.

Conventional digital VLSI is a mature and therefore reliable technology. NNs implemented using this technology are based on simple processor array architectures with either a processor-per-synapse [2] [5] or a processor-per-neuron organization [10] [11]. The attractions of using this technology to implement neural systems are that precision can be arbitrary specified, it is possible to implement any training/learning algorithm, and the effects of scaling up an implementation are predictable. Digital implementations are larger and slower than analogue implementations nevertheless they represent the most reliable and lowest risk path from high level simulation to hardware implementation of a real-time NN.

In this paper a 2-D Systolic Array (SA) of dedicated Processing Elements (PEs) named also Systolic Cells (SCs) is presented as the heart of a Hopfield Neural Network accelerator. The 2-D SA that we propose in this paper has a processor-per-synapse architecture; each SC holds an element of the synaptic weight matrix. The resulting SC structure is simple, and the 2-D SA has a regular architecture with identical SCs disposed in a mesh-connected array. This regular architecture is suitable for VLSI implementations. A FPGA implementation of the resulting SC is presented. The major advantage of the FPGA based implementation is the dynamic reconfigurability.

2 The Hopfield Network

The Hopfield model [14] is a feedback network in which each neuron has synaptic interconnections with all the other neurons. There are not input and output specialized units in

0-7803-7044-9/01/$10.00 ©2001 IEEE

this architecture; each neuron is in the same time input and output and what it is important for the network is his state. Hopfield network is a dynamical system and his dynamics is characterized by the evolution of the neurons state in time. The target is to exploit this natural dynamics to determine the network to work like an associative memory. In the discrete form the Hopfield network dynamics can be described as follows:

$$p_i(t) = \sum_{j=1}^{N} w_{ij} \cdot x_j(t-1) \qquad (1)$$

$$x_i(t) = \sigma(p_i(t) - 2\theta_i) \qquad (2)$$

where w_{ij} represents a coefficient or synaptic weight associated with the j-th input x_j and the i-th neuron. The weighted sum p_i is called potential and σ represents the activation function. That is, the recall phase of the Hopfield network consists in a recurrence of following form:

$$\mathbf{x}^{[k]} = \sigma \cdot (\mathbf{W} \cdot \mathbf{x}^{[k-1]}); \quad k=1,2,...,N \qquad (3)$$

The convergence occur when $\mathbf{x}^{[N]} = \mathbf{x}^{[N-1]}$. Therefore, $\mathbf{x}^{[N]}$ represents the response (output vector) of the Hopfield network (characterized by the weight matrix W) to the input vector $\mathbf{x}^{[0]}$.

3 The 2-D Systolic Architecture with Fixed Weights

The proposed 2-D SA is a processor-per-synapse architecture. For an N-components input vector x, the W matrix will have N^2 weights, which correspond to the N^2 Systolic Cells (SCs) in the 2-D SA. The input vector components are loaded into the SA from north and cross the array to south. The partial sums (PSs) cross the SA from west to east and the output vector components are obtained on the east board of the SA. The moving data are:

- the x vector components which cross the SA from north to south
- the partial sums PSs which cross the SA from west to east.

Each SC (i,j) receives the x_j component of the x vector at time t, computes the local product $w_{ij} \cdot x_j$ and adds this product to the partial sum PS_i received from west. The new calculated PS_i is sent to the neighboring SC on east. We will call recurrence step the time for computing the $\mathbf{W} \cdot \mathbf{x}$ multiplication.

When an output vector component y_i comes out on the east board of the SA, the activation function σ will be applied. σ function will be not integrated inside SA; specialized units placed on the east board of the SA will compute it. The resulting vector y will be the input vector for the new recurrence and must be reinserted from north into the SA. Being not local data transmission, this loop destroys the homogeneity of the data transmissions and can generate increased propagation delays, which will affect the global performances of the SA. It is preferable to convert this loop into local data transmissions to the nearest neighbor (SC).

4 An Architecture with Local Data Transmissions

This architecture is based on 3 rules:

r1: Systolic sequencing of the operations inside the SA.

r2: Each SC communicates only with the four nearest neighboring SCs.

r3: At the end of a recurrence the output vector will be in the same initial position in which was the input vector at the beginning of the recurrence; that is, the output (new input) vector is ready to start a new recurrence.

The basic idea refers to the initial position of the input vector, which will be placed on the main diagonal of the SA (figure 2). All the SCs will have identical internal structures (figure 1) and will communicate data with only the four nearest neighboring cells.

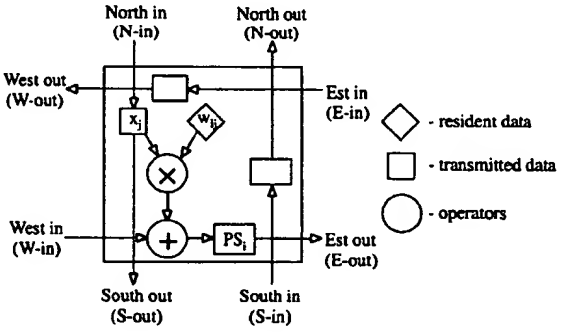

Figure 1: Logic diagram of the SC

5 The Systolic Algorithm

The algorithm for computing an output vector (a recurrence step) can be divided into 3 successive phases:

- gradual computing of the partial sums.
- applying the activation function σ.
- positioning the output (new input) vector for a new recurrence.

We will present a recurrence step for a 3×3 weight matrix.

5.1 First Stage

How already stated, the input vector **x** will be placed in the main diagonal of the SA. Complying with **r1** rule we will compute the first product (figure 2). If we note:

$PS_{i,j}$ – the partial sum in the SC(i,j).
$x_{i,r}$ – the i-th component of the input vector **x** during the recurrence step r.

the SA will compute the first partial sum:

$$PS_{1,1} = w_{11} \cdot x_{1,1} \quad (4)$$

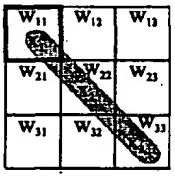

Figure 2: First stage (r-recurrence step)

5.2 Second Stage

During the second stage (figure 3) will compute the w_{21} and w_{12} cells. The data will be transmitted inside the SA so that will be applied to the mentioned SCs:

- the weights (**W** matrix components) are resident data.
- the input vector components $x_{i,1}$ will be propagated to the north and will be reflected successively on the north board of the SA.
- the partial sum $PS_{1,1}$ will be transmitted towards east.

After these local data transmissions, the local products will be calculated and will be accumulated to the partial sums, which enter into the two SCs from west:

$$PS_{1,2} = PS_{1,1} + w_{12} \cdot x_{2,1} \quad (5)$$
$$PS_{2,1} = w_{21} \cdot x_{1,1} \quad (6)$$

At the end of this stage, the partial sums $PS_{1,2}$ and $PS_{2,1}$ are located into the w_{12} and w_{21} SCs.

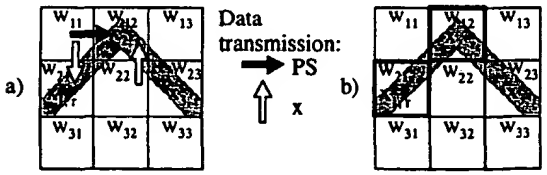

Figure 3: Second stage
a) data communications b) calculus

5.3 Third Stage

The mobile data will be transmitted like in the previous stage. The w_{13}, w_{22} and w_{31} cells will compute (figure 4):

$$PS_{1,3} = PS_{1,2} + w_{13} \cdot x_{3,1} \quad (7)$$
$$PS_{2,2} = PS_{2,1} + w_{22} \cdot x_{2,1} \quad (8)$$
$$PS_{3,1} = w_{31} \cdot x_{1,1} \quad (9)$$

At the end of the third stage, the first component of the output vector (the first final sum) will be stored in the w_{13} SC.

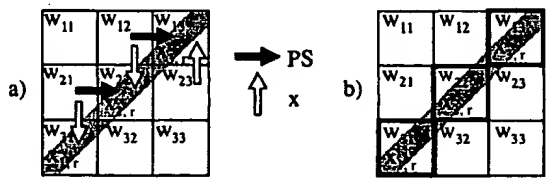

Figure 4: Third stage
a) data communications b) calculus

5.4 Fourth Stage

The first component is available on the east board of the SA and will be processed applying the activation function σ. That is, the first component of the new input vector will be generated:

$$x_{1,2} = \sigma(PS_{1,3}) \quad (10)$$

The mobile data will be transmitted like in figure 5 and the w_{23} and w_{32} cells will compute:

$$PS_{2,3} = PS_{2,2} + w_{23} \cdot x_{3,1} \quad (11)$$
$$PS_{3,2} = PS_{3,1} + w_{32} \cdot x_{2,1} \quad (12)$$

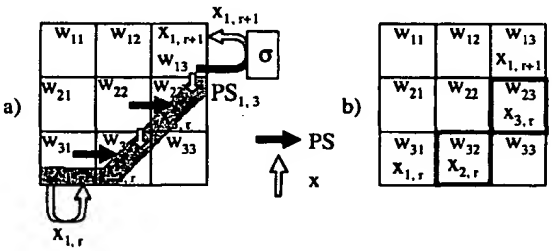

Figure 5: Fourth stage - applying the activation function σ and the retransmission towards the main diagonal of the $x_{1,r+1}$ component

5.5 Fifth Stage

It is similar to the previous stage; the second component of the resulting vector will be retransmitted towards the main diagonal of the SA and the last final sum will be calculated:

$$x_{2,2} = \sigma(PS_{2,3}) \quad (13)$$

$$PS_{3,3} = PS_{3,2} + w_{33} \cdot x_{3,1} \quad (14)$$

5.6 Sixth Stage

In the sixth stage the **W·x** multiplication will be completed, the activation function σ will be applied to the last component of the resulting vector and the resulting vector will be placed on the main diagonal of the SA (figure 6):

$$x_{3,2} = \sigma(PS_{3,3}) \quad (15)$$

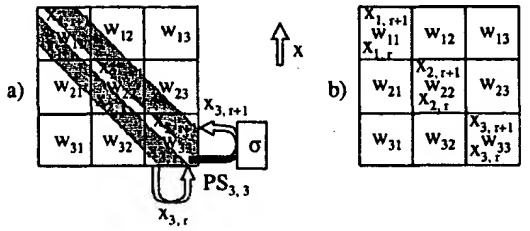

Figure 6: Sixth stage
a) data communications b) calculus

At the end of the sixth stage, both the new $\mathbf{x}_{r+1}$ vector components and the old $\mathbf{x}_r$ vector components are located in the main diagonal of the SA. The **r1**, **r2** and **r3** rules where complied and the SA is ready for a new recurrence. The presence of both $\mathbf{x}_r$ and $\mathbf{x}_{r+1}$ vectors in the main diagonal of the SA permits the convergence detection of the Hopfield algorithm (comparing the two vectors). Comparing the two components $x_{i,r}$ and $x_{i,r+1}$ will give a local result connected to the convergence detection. The global convergence signal, for a N×N SA, will be computed as in (16) and systolically generated (figure 7).

$$Conv(N) = \prod_{i=1}^{N} (x_{i,r} = x_{i,r+1}) = \prod_{i=1}^{N} CV_i \quad (16)$$

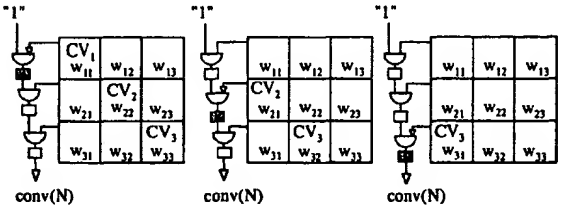

Figure 7: Systolic propagation of the convergence signal

In order to increase the 2-D SA performances, the convergence detection for the current recurrence step and the output vector computing for the next recurrence step will be overlapped systolic operations.

6 FPGA implementation of the 2-D SA

In order to evaluate the systolic implementation of different NNs architectures we designed also the 2-D SA (practically the SC structure) for other neural algorithms (i.e. SOFM algorithm). Using the XILINX software, we evaluated, also, the implantability of the 2-D SA into the FPGA structures; we designed and successfully tested the SC block. The FPGA based implementation is feasibly (i.e. XC 40250 FPGA chip, which contains 20000 CLBs can hold a 22×22 SA [5]. A speed-up board with 36 XC 40250 chips can hold a 132×132 SA.

The major advantage of the FPGA based implementation is the dynamic reconfigurability. This implementation may be used to build a speed-up hardware integrated in a host computer. The speed-up hardware will not execute any time all neural algorithms. In order to keep a very simple SC structure, it is convenient to classify the neural algorithms into classes (based on the resemblance between different algorithms). Each class will have a dedicated SC structure. Therefore, the SC structure will be different from a class to another but all SCs will have a common feature: simplicity and, therefore, high processing speed at hardware level. Depending on the neural application issued by the user, the software driver of the neuroprocessor board will automatically configure the SC structure by programming the FPGA chips. Therefore, we can obtain maximum processing speed at hardware level for each neural algorithm.

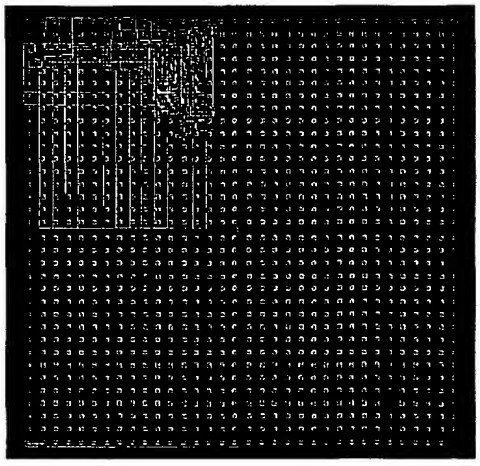

Figure 8: The SC implemented in the XC 4025E chip

7 Performance

The two-dimensional SA proposed in this paper represents an implementation that integrates synapses; integration of

synapses facilitates network extension and input/output communications. Although no practical performance measurements have been realized so far, a theoretical peak performance can be calculated using the two metrics for neurocomputers:

1. number of *connections per second* (CPS), for recall phase
2. number of *connection updates per second* (CUPS), for learning phase

The expected peak performance of the 2-D SA is given by:

$$P = \frac{N^2 \cdot f}{n_{op} \cdot N_{PS}} \cdot U \qquad (17)$$

where: N^2 - total number of SCs
f - clock frequency
n_{op} - number of operations required to compute a connection (recall phase) or to update it (learning phase)
N_{PS} - number of clock cycles per operation
U - utilization rate of SA

Considering the Hopfield network (n_{op}=1) implemented in the 132×132 SA (36 XC40250 FPGA chips) working at a clock frequency of f=100MHz, with N_{PS}=64 (serial communications of 64 bits integers), the peak performance (U=100%) will be:

$$P = \frac{132^2 \cdot 100 \cdot 10^6}{1 \cdot 64} \cdot 100\% = 27.225\,\text{GCUPS}$$

This value compare well to those reported for supercomputers or other NN dedicated systems.

The square W matrix of the Hopfield NN fits well with the square SA. Other NN architectures (including multilayer feedforward NN) have rectangle weight matrices and is necessary to add zero elements in order to arrive to a square matrix that fit with the square SA. The utilization rate U can be less than 100% but in most of the cases is bigger than 70%. The described architecture represents a good compromise between cost and performance, between simplicity and regularity of design (thus implying a reduced cost) and generality of use.

7 Conclusions

There are a variety of models in the field of NNs, which differ by structure and by the performed functions. The choice of the appropriate NN architecture, for a defined application, can't be based on well-defined criterions. Therefore, the research activity in the field of NN requires more and more efficient tools. Speed-up hardware integrated in a host system, like a workstation, can be a good compromise between cost and performance. The SA presented in this paper represents our proposition for this speed-up hardware. The advantages of the proposed systolic architecture can be resumed as follows:

- **Modularity**. It can be detected at different levels:
  - The SC is built by using three types of functional blocks: register, adder and multiplexor.
  - The VLSI chip contains a mesh-connected matrix of identical SCs.
  - The SA, in whole, will be built like a mesh-connected chip array, with interconnections between nearest neighbor chips.
- **Extensibility**. The SA can be extended in a very simple way by adding new FPGA chips to the existent mesh-connected array of FPGA chips.
- **Simplicity**. SC has a very simple structure, based on some registers and operators. Therefore, the SC design, test and performance evaluation processes can be fast and sure.
- **Reconfigurability**. Programming FPGA chips can configure the 2-D SA. Software driver can do this in a dynamic way, depending on the neural algorithm issued by the user.
- **Fine-grain parallelism**. The 2-D SA presented in this paper integrates synapses and not neurons. It is a fine-grain parallelism, which is much more suitable with the distributed structure of NNs; NNs are high parallel, distributed and fault tolerant systems.

8 References

[1] Mihu I. Z., "Systolic neuroprocessors – from concepts to implementation", *Computer Science Education Challenges for the New Millenium*, Gerrit C. van der Veer, Ioan Alfred Leţia Editors, Ed. Casa Cartii de Stiinta, Cluj-Napoca, Romania, 1999, pp. 5 – 25.

[2] Mihu I. Z., "Systolic implementation analysis of Hopfield neural algorithm", *Acta Univarsitatis Cibiniensis (Issued for BEYOND 2000 International Conference)*, vol. XXXVIII, Sibiu, Romania, 1999, pp. 85 – 92.

[3] Lehmann C., Viredaz M. A., Blayo F., "A generic systolic array building block for neural networks with on - chip learning", *IEEE Trans. on Neural Networks*, vol. 4, no. 3, 1993, pp. 400 - 407.

[4] Maria N., Dugué A.G., Blayo F., "1D and 2D Systolic Implementations for Radial Basis Function Networks", *IEEE Proc. of MicroNeuro*, 1994, pp. 34 - 45.

[5] Mihu I. Z., "Solutions of Neural Networks implementation on Systolic Architectures", *Ph.D. Thesis*, Technical University of Timisoara, 1998.

[6] Johnson K. T., Hurson A. R., "General - purpose systolic arrays", *IEEE Comp.*, November 1993, pp. 20 - 31.

[7] Ienne P., Viredaz M. A, "GENES IV: A bit-serial processing element for multimodel neural-network accelerator", *Journal VLSI Signal Processing*, vol. 9, nr. 3, 1995, pp. 257 - 273.

[8] Cornu T., Ienne P., Niebur D., Thiran P., Viredaz M. A., "Design, implementation, and test of a multimodel systolic neural-network accelerator", *Scientific Programming*, vol. 5, nr.1, 1996, pp. 47 - 61.

[9] Fortes J. A. B., Wah B. W., "Systolic Arrays: from concepts to implementation", *IEEE Computer*, 1987.

[10] Naylor D., Jones S., Myers D., "Backpropagation in Linear Arrays – A Performance Analysis and Optimization", *IEEE Trans. on Neural Networks*, Vol. 6, no. 3, 1995, pp. 583-595.

[11] Ramacher U., "SYNAPSE - A neurocomputer that synthesizes neural algorithms on a parallel systolic engine". *Journal of Parallel and Distributed Comput.*, vol. 14, nr. 3, 1992, pp. 306 - 318.

[12] Shen B.J., Choi J., *Neural Information Processing and VLSI*, Kluwer Academic Publishers, Boston, 1995.

[13] Viredaz M. A., "MANTRA I: An SIMD processor array for neural computation", *Proceedings of Euro - ARCH '93*, Munich, Germany, 1993.

[14] Zurada J. M., *Introduction to Artificial Neural Systems*, West Publishing Company, St. Paul, 1992.